MetAug: Contrastive Learning via Meta Feature Augmentation

Jiangmeng Li*
Institute of Software Chinese Academy of Sciences
University of Chinese Academy of Sciences
jiangmeng2019@iscas.ac.cn

Wenwen Qiang*
Institute of Software Chinese Academy of Sciences
University of Chinese Academy of Sciences
a01114115@163.com

Bing Su
Renmin University of China
subingats@gmail.com

Changwen Zheng
Institute of Software Chinese Academy of Sciences
changwen@iscas.ac.cn

Hui Xiong
Artificial Intelligence Thrust, The Hong Kong University of Science and Technology
xionghui@ust.hk

Abstract

What matters for contrastive learning? We argue that contrastive learning heavily relies on informative features, or "hard" (positive or negative) features. Early works include more informative features by applying complex data augmentations and large batch size or memory bank, and recent works design elaborate sampling approaches to explore informative features. The key challenge toward exploring such features is that the source multi-view data is generated by applying random data augmentations, making it infeasible to always add useful information in the augmented data. Consequently, the informativeness of features learned from such augmented data is limited. In response, we propose to directly augment the features in latent space, thereby learning discriminative representations without a large amount of input data. We perform a meta learning technique to build the augmentation generator that updates its network parameters by considering the performance of the encoder. However, insufficient input data may lead the encoder to learn collapsed features and therefore malfunction the augmentation generator. A new margin-injected regularization is further added in the objective function to avoid the encoder learning a degenerate mapping. To contrast all features in one gradient back-propagation step, we adopt the proposed optimization-driven unified contrastive loss instead of the conventional contrastive loss. Empirically, our method achieves state-of-the-art results on several benchmark datasets.

*They have contributed equally to this work.

1. Introduction

Contrastive learning methods have achieved empirical success in computer vision [10, 22]. Under the setting of self-supervised learning (SSL), recent researches demonstrate the superiority of contrastive methods [11, 25, 35, 42]. Typically, these approaches learn features by contrasting different views (e.g., different random data augmentations) of the image in hidden space. We recap the preliminaries of the conventional contrastive learning paradigm: every two views of the *same* image are considered to be a positive pair, and every two views of the *different* images are considered to be a negative pair; the contrastive loss [21, 34] guides the learned features to bring *positive* pairs together and push *negative* pairs farther apart.

However, this learning paradigm suffers from the need for a large number of pairs to contrast, e.g., large batch size or memory bank size, because many pairs are not informative to the model, i.e., positive pairs are pretty close and negative pairs are already very apart in hidden space. These pairs have few contributions to the optimization. Contrastive methods need numerous pairs and expect to collect informative ones, and therefore complex data augmentations (e.g., jittering,

random cropping, separating color channels, etc.) [3, 7] and large-scale memory banks [23, 42] are effective in improving the performance of contrastive models on downstream tasks.

The success of the recent works depends on the elaborate selection of informative negative pairs [11, 35]. These methods focus on designing sampling strategies to assign larger weights to informative pairs, which rely on enough and informative positive pairs and do not need large amounts of negative pairs. When the number of pairs to contrast is limited, the contrastive loss may cause conventional contrastive learning approaches to learn collapsed features [20, 51], e.g., outputting the same feature vector for all images.

Nowadays, many researchers have noticed potential environmental problems brought by training deep learning models [50], for instance, [40] reports a remarkable example that the carbon dioxide emissions generated by training a Transformer [45] is equivalent to 200 round trips between San Francisco and New York by plane. Therefore, we motivate our method to perform an efficient self-supervised contrastive approach to learn anti-collapse and discriminative features based on a restricted amount of images in a training epoch (e.g., small batch size) and plain neural networks with limited parameters. Much research effort has been devoted to strong augmentations on *data*, but the informativeness of the features learned from the augmented data is hard to exactly measure, since the data is fed into mapping-agnostic deep neural networks to generate the features. Instead, we directly tackle augmentations on features and show that appropriate feature augmentations can sharply improve the optimization.

To this end, we propose ***Met****a Feature* ***Aug****mentation* (MetAug), which learns view-specific encoders (with projection heads) and auxiliary meta feature augmentation generators (MAGs) by margin-injected meta feature augmentation and optimization-driven unified contrast. Suppose the input data has $\boldsymbol{M}$ views, and the multi-view data is fed into the encoder to generate the latent features. We initialize $\boldsymbol{M}$ neural networks as MAGs for views, which are used to augment the features of each view. We contrast all original and augmented features for bi-optimization training. Through such a learning paradigm, MetAug can improve the performance of self-supervised contrastive learning.

To learn anti-collapse and discriminative features from a restricted amount of images, MetAug relies on two key ingredients: 1) margin-injected meta feature augmentation, where MAGs use the performance of the encoder in one iteration to improve the view-specific feature augmentations for the next iteration. In this way, MAGs promote the encoder to efficiently explore the discriminative information of the input. For the original features and the augmented features generated by MAGs, we inject a margin $\mathcal{R}_\sigma$ between the similarities of them, which avoids the instance-level feature collapse; 2) optimization-driven unified contrast, which contrasts all features in one gradient back-propagation step. Such proposed contrast can also amplify the impact of the instance similarity that deviates far from the optimum and weaken the impact of the instance similarity that is close to the optimum. We conduct head-to-head comparisons on various benchmark datasets, which prove the effectiveness of margin-injected meta feature augmentation and optimization-driven unified contrast. **Contributions**:

- We propose margin-injected meta feature augmentation, which directly augments the latent features to generate informative and anti-collapse features. Benefiting from such features, encoders can efficiently capture discriminative information.
- We propose optimization-driven unified contrast to include all available features in one step of back-propagation and weight the similarities of paired features by measuring their contributions to optimization.
- Empirically, MetAug improves the downstream task performance on different benchmark datasets.

2. Related works

Self-supervised learning. Under the setting of unsupervised learning, SSL methods have achieved impressive success, which constructs auxiliary tasks to learn discriminative information from the unlabeled inputs. Deep InfoMax [25] explores to maximize the mutual information between an input and the output of a deep neural network encoder by different mutual information estimations. CPC [34] proposes to adopt noise-contrastive estimation (NCE) [21] as the contrastive loss to train the model to measure the mutual information of multiple views deduced by the Kullback-Leibler divergence [19]. CMC [42] and AMDIM [3] employ contrastive learning on the multi-view data. SwAV [6] compares the cluster assignments under different views instead of directly comparing features by using more views (e.g., six views). SimCLR [7] and MoCo [23] use large batch or memory bank to enlarge the amount of available negative features to learn good representations. Instead of exploring informative features by adopting various data augmentations and enlarging the number of features, our method focuses on straightforwardly generating informative features to contrast.

Recent works explore imposing stronger constraints on the conventional contrastive learning paradigm or propose alternative loss functions (instead of contrastive loss). DebiasedCL [11] and HardCL [35] consider to directly collect informative features to contrast by designing sampling strategies, which are inspired by positive-unlabeled learning methods [14, 16]. Motivated by [39], [44] proposes an information theoretical framework for SSL, which, guided by the theory, uses information bottleneck to restrict the learned features and maintain the sufficient self-supervision.

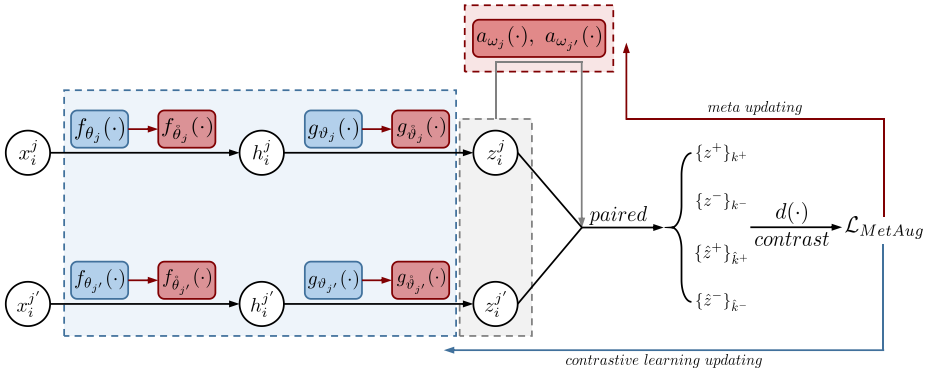


Figure 1. MetAug's architecture. *Dashed blue* box represents the data encoding process, and *dashed red* box represents the meta feature augmentation. In training, we first fix a_{ω_j} and $a_{\omega_{j'}}$, and then train $f_{\theta_j}(\cdot)$, $g_{\vartheta_j}(\cdot)$, $f_{\theta_{j'}}(\cdot)$ and $g_{\vartheta_{j'}}(\cdot)$ by using $\mathcal{L}_{MetAug}$. Next, we fix the encoders and projection heads, and train a_{ω_j} and $a_{\omega_{j'}}$ in a meta updating manner. The networks are iteratively trained until convergence.

BYOL [20], W-MSE [17], and Barlow Twins [51] present a crucial issue that insufficient self-supervision (e.g., not enough negative features) may lead to the feature collapse in hidden space. To tackle the mentioned issue, we propose a new margin-injected regularization in meta feature augmentation to avoid generating degenerate features. DACL [46] proposes a new data augmentation that applies to domain-agnostic problems. LooC [49] learns to capture varying and invariant factors for visual representations by constructing separate embedding spaces for each augmentation. These methods explore informative features from the perspective of data augmentation, while our straightforward idea behind our method is to augment features in the latent space.

Meta learning. The objective of meta learning is to automatically learn the *learning algorithm*. Early works [1,4,36] aim to guide the model (e.g., neural network) to learn prior knowledge about *how to learn new knowledge*, so that the model can efficiently learn new knowledge, e.g., the model can be quickly fine-tuned to specific downstream tasks with few training steps and achieve good performance. Recently, researchers explored to use meta learning to find optimal hyper-parameters [31] and appropriately initialize a neural network for few-shot learning [18, 38, 47]. Recent approaches [9, 27, 32, 33] have focused on learning optimizers or generating a gradient-driven loss for deep neural networks in the field of NLP, computer vision, etc.

3. Method

Our goal is to learn representations that capture information shared between multiple different views by performing self-supervised contrastive learning. Formally, we suppose the input multi-view dataset as $X = \{X_1, X_2, ..., X_N\}$, where N denotes the number of samples. X_i represents a collection of M views of the sample, where $i \in \{1, ..., N\}$. For each sample X_i, we denote x_i as a random variable representing views following $x_i \sim \mathcal{P}(X_i)$, and x_i^j denotes the j-th view of the i-th sample, where $j \in \{1, ..., M\}$.

3.1. Contrastive learning preliminary

We recap the preliminaries of contrastive learning [7, 42]: the foundational idea behind contrastive learning is to learn an embedding that maximizes agreement between the views of the same sample and separates the views of different samples in latent space. Given a multi-view dataset X, we treat pairs of the views of the same sample $\{x_i^j, x_i^{j'}\}$, where $j, j' \in \{1, ..., M\}$, as *positives*, versus pairs of the views of the different samples $\{x_i^j, x_{i'}^{j'}\}$, where $i \neq i'$, as *negatives*. To impose contrastive learning, we feed the input x_i^j into a view-specific encoder $f_{\theta_j}(\cdot)$ to learn a representation h_i^j, and h_i^j is mapped into a feature z_i^j by a projection head $g_{\vartheta_j}(\cdot)$, where θ_j and ϑ_j are the network parameters of $f_{\theta_j}(\cdot)$ and $g_{\vartheta_j}(\cdot)$, respectively. A discriminating function $d(\cdot)$ is adopted to measure the similarity of $\{z_i^j, z_{i'}^{j'}\}$, where $i \neq i'$. The encoder $f_{\theta_j}(\cdot)$ and projection head $g_{\vartheta_j}(\cdot)$ are trained by using a contrastive loss [34], which is formulated as follows:

$$\mathcal{L} = -\mathop{\mathbb{E}}_{X_S}\left[\log \frac{d\left(\{z^+\}\right)}{d\left(\{z^+\}\right) + \sum_{k=1}^{K} d\left(\{z^-\}_k\right)}\right] \tag{1}$$

where $X_S = \{\{z^+\}, \{z^-\}_1, \{z^-\}_2, ..., , \{z^-\}_k\}$ is a set of *pairs* randomly sampled from X, which includes a positive $\{z^+\}$ and K negatives $\{z^-\}_k$, $k \in \{1, ..., K\}$, because contrastive loss can only use *one* positive in an iteration. In test, the projection head $g_{\vartheta_j}(\cdot)$ is discarded, and the representation h_i^j is directly used for downstream tasks.

3.2. Margin-injected meta feature augmentation

Recent contrastive methods rely on complex data augmentations to increase the informativeness of views. Yet this lack of guidance approach leads to the demand for a large number of training data (e.g., large batch size and memory bank). We propose a meta feature augmentation method, which creates informative augmented features by updating

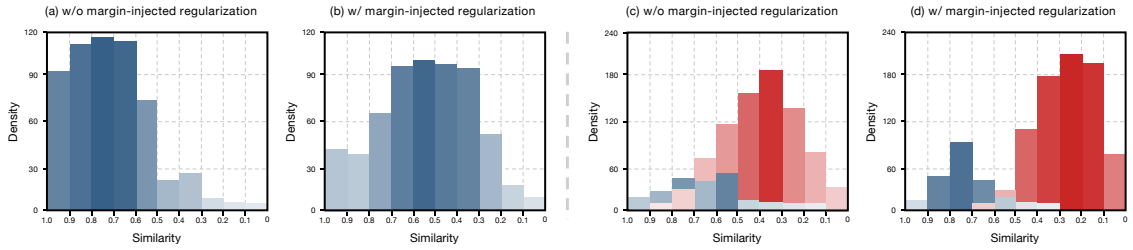


Figure 2. Similarity histograms obtained by our method (with or without margin-injected regularization) on CIFAR-10. (a) and (b) demonstrate the summarized similarity of positives (i.e., $\{\hat{z}^+\}$) that include original features and augmented features. (c) and (d) demonstrate the statistical results of the original features learned by our model. *Blue* histograms represent the similarity between the features of the same image's views, and *red* histograms represent the similarity between the features of the different images' views.

parameters of its own network according to the performance (gradient) of the encoder (see Appendix A.3 for our rethinking of augmented features). A visualization of the overall MetAug architecture is shown in Figure 1.

To this end, we build a group of MAGs $a_\omega(\cdot) = \{a_{\omega_1}(\cdot), ..., a_{\omega_M}(\cdot)\}$ for all M views, where $\omega = \{\omega_1, ..., \omega_M\}$. To be simplified, we define f_θ and g_ϑ as the groups of view-specific encoders and projection heads, respectively, i.e., $\theta = \{\theta_1, ..., \theta_M\}$ and $\vartheta = \{\vartheta_1, ..., \vartheta_M\}$.

In training, the encoders $f_\theta(\cdot)$ and the projection heads $g_\vartheta(\cdot)$ are trained alongside the MAG $a_\omega(\cdot)$ (with the network parameters ω). Following the protocol of meta learning [18, 32], we firstly train $f_\theta(\cdot)$ and $g_\vartheta(\cdot)$ under the learning paradigm of self-supervised contrastive learning. Then, $a_\omega(\cdot)$ is updated by computing its gradients with respect to the performance of $f_\theta(\cdot)$ and $g_\vartheta(\cdot)$. Here, we measure the performance of $f_\theta(\cdot)$ and $g_\vartheta(\cdot)$ by the gradients of them when the corresponding contrastive loss is back-propagated. Concretely, all of $f_\theta(\cdot)$, $g_\vartheta(\cdot)$, and $a_\omega(\cdot)$ are iteratively trained until convergence.

Specifically, we first update network parameters θ and ϑ of the encoders and projection heads by adopting the conventional contrastive loss. Then, we train the MAG $a_\omega(\cdot)$ in a meta learning manner. We encourage the augmented features to be informative, and the encoders $f_\theta(\cdot)$ can better explore the discriminative information by jointly using the original and augmented features to contrast. Hence, the performance of the encoders would be promoted on the same training data. To update network parameters ω of the $a_\omega(\cdot)$, we formalize the meta updating objective as follows:

$$\arg\min_{\omega} \left(\mathcal{L}\left(\left\{ g_{\mathring{\vartheta}}\left(f_{\mathring{\theta}}(\widetilde{X})\right), a_\omega\left(g_{\mathring{\vartheta}}\left(f_{\mathring{\theta}}(\widetilde{X})\right)\right)\right\}\right)\right) \quad (2)$$

where $\widetilde{X}$ represents a minibatch sampled from the training dataset X, $\left\{ g_{\mathring{\vartheta}}\left(f_{\mathring{\theta}}(\widetilde{X})\right), a_\omega\left(g_{\mathring{\vartheta}}\left(f_{\mathring{\theta}}(\widetilde{X})\right)\right)\right\}$ denotes a set including both original features and meta augmented features. $\mathring{\vartheta}$ and $\mathring{\theta}$ represent the parameter sets of the encoders and projection heads, respectively, which are computed by the updating of one gradient back-propagation using the contrastive loss:

$$\begin{aligned}
\mathring{\theta} &= \theta - \ell \cdot \Delta_\theta \left(\mathcal{L}\left(\left\{ g_\vartheta\left(f_\theta(\widetilde{X})\right), a_\omega\left(g_\vartheta\left(f_\theta(\widetilde{X})\right)\right)\right\}\right)\right) \\
\mathring{\vartheta} &= \vartheta - \ell \cdot \Delta_\vartheta \left(\mathcal{L}\left(\left\{ g_\vartheta\left(f_\theta(\widetilde{X})\right), a_\omega\left(g_\vartheta\left(f_\theta(\widetilde{X})\right)\right)\right\}\right)\right)
\end{aligned} \quad (3)$$

where ℓ is the learning rate shared between θ and ϑ. The idea behind the meta updating objective is that we perform the second-derivative technique [18, 32, 53] to train $a_\omega(\cdot)$. Specifically, a derivative over the derivative (Hessian matrix) of the combination $\{\theta, \vartheta\}$ is used to update ω, where $\{\theta, \vartheta\}$ is a parameter set conjoining θ and ϑ. We compute the derivative with respect to ω by using a retained computational graph of $\{\theta, \vartheta\}$.

However, in practice, we find a critical issue: when the original features are not informative enough, large gradients are difficult to generate by contrasting the uninformative features, the MAGs $a_\omega(\cdot)$ are inclined to create collapsed augmented features, e.g., the augmented features and the original features are very similar. We consider the reason for the feature collapse is that small gradient changes of the encoders alongside projection heads $g_\vartheta(f_\theta(\cdot))$ lead to the update step-size of $a_\omega(\cdot)$ to become extensively small, which leaves the optimization of $a_\omega(\cdot)$ to fall into a local optimum. The augmented features are such that without any extra useful information. To tackle this issue, we further inject a margin to encourage $a_\omega(\cdot)$ to generate more complex and informative augmented features, which can be considered as a regularization term in the meta updating objective. See Figure 2(a) for the details of the augmented feature collapse issue, and we observe that, without margin-injected regularization, MAGs tend to generate collapsed features that are very similar with the original features. Formally, we formulate the approach to generate margins for $a_\omega(\cdot)$ by

$$\begin{aligned}
\sigma^+ &= \min\left[\min\left(\left\{ d(\{z^+\}_{k^+})\right\}\right), \max\left(\left\{ d(\{z^-\}_{k^-})\right\}\right)\right] \\
\sigma^- &= \max\left[\min\left(\left\{ d(\{z^+\}_{k^+})\right\}\right), \max\left(\left\{ d(\{z^-\}_{k^-})\right\}\right)\right]
\end{aligned} \quad (4)$$

where $\left\{d(\{z^+\}_{k^+})\right\}$ is a set of the outputs (similarities) of positives computed by the discriminating function $d(\cdot)$, and $k^+ \in \{1, ..., K^+\}$ where K^+ represents the number of positives in a minibatch. $\left\{d(\{z^-\}_{k^-})\right\}$ is a set of the discriminating outputs of negatives, and $k^- \in \{1, ..., K^-\}$ where K^- represents the number of negatives. Note that only original features are used in Equation 4. We call the formulated margin generation approach as *"Large"*, and we also propose two more approaches, called *"Medium"* and *"Small"*. In Appendix A.2, we conduct comparisons to evaluate the effects of the three margin generation approaches. We inject the margins between the augmented features and original features by adding a regularization term in the meta updating objective, and the regularization is defined as:

$$\begin{aligned}\mathcal{R}_\sigma =& \frac{1}{\hat{K}^+}\sum_{\hat{k}^+=1}^{\hat{K}^+}\left[d(\{\hat{z}^+\}_{\hat{k}^+}) - \sigma^+\right]_+ \\ &+ \frac{1}{\hat{K}^-}\sum_{\hat{k}^-=1}^{\hat{K}^-}\left[\sigma^- - d(\{\hat{z}^-\}_{\hat{k}^-})\right]_+\end{aligned} \tag{5}$$

where $\{\hat{z}^+\}_{\hat{k}^+}$ denotes a positive that includes one *original* feature and one *augmented* feature, and $\hat{K}^+$ denotes the number of such positives. $\{\hat{z}^-\}_{\hat{k}^-}$ represents likewise one of $\hat{K}^-$ negatives, each of which includes one *original* feature and one *augmented* feature. $[\cdot]_+$ denotes the cut-off-at-zero function, which is defined as $[a]_+ = \max(a, 0)$. We then integrate such regularization to the updating of ω by

$$\begin{aligned}\omega \leftarrow \omega - \ell' \cdot \Delta_\omega\Bigg(&\mathcal{L}\Big(\Big\{g_{\hat{\vartheta}}(f_{\hat{\theta}}(\widetilde{X})), \\ &a_\omega\Big(g_{\hat{\vartheta}}(f_{\hat{\theta}}(\widetilde{X}))\Big)\Big\}\Big) + \alpha \cdot \mathcal{R}_\sigma\Bigg)\end{aligned} \tag{6}$$

where ℓ' represents the learning rate of ω, and α is a hyperparameter balancing the impact of the loss of margin-injected regularization term. $\mathcal{R}_\sigma$ restricts MAGs to generate informative features that are more different with the original features (see Figure 2(b)). In practice, Figure 2(c) and (d) show that the features learned by our method (with margin-injected regularization) are more concentrated, e.g., the features of the same image are more similar and the gap between the features of the different images are enlarged, which proves informative augmented features can further lead the encoders to learn non-collapsed (scattered) features.

3.3. Optimization-driven unified contrast

We propose to jointly contrast all features (including the original features and the meta augmented features) in one gradient back-propagation step. Motivated by [37], we introduce the following optimization-driven unified loss function to replace the conventional contrastive loss as follows:

$$\mathcal{L}_{OUCL} = \left[\sum_{k^-=1}^{K^-} d(\{z^-\}_{k^-}) - \sum_{k^+=1}^{K^+} d(\{z^+\}_{k^+}) + \lambda\right]_+ \tag{7}$$

where $[\cdot]_+$ ensures that $\mathcal{L} \geq 0$ is always held. Note that all original features and augmented features are involved. λ is a margin between the summarized instance similarities to enhance the capability of the similarity separation. However, we find that the difference between $\sum_{k^-=1}^{K^-} d(\{z^-\}_{k^-})$ and $\sum_{k^+=1}^{K^+} d(\{z^+\}_{k^+})$ is not the larger the better. Excessive increases of the difference may undermine the convergence in optimization. We thereby wish to adopt a margin λ that leads to preferable convergence. We reform the loss in Equation 7 by adding a temperature coefficient β as follows:

$$\begin{aligned}\mathcal{L}_{OUCL} =& \frac{1}{\beta} log\Bigg\{1 + \sum_{k^-=1}^{K^-}\sum_{k^+=1}^{K^+} \\ &exp\Big[\beta\Big(d(\{z^-\}_{k^-}) - d(\{z^+\}_{k^+}) + \lambda\Big)\Big]\Bigg\}\end{aligned} \tag{8}$$

when $\beta \to +\infty$, Equation 8 is Equation 7. Inspired by [41], we use weighting factors Γ^- and Γ^+ to modulate the impacts of $d(\{z^-\}_{k^-})$ and $d(\{z^+\}_{k^+})$. Such approach aims to give greater weight to the similarity that deviates from the optimum and smaller weight to the similarity that has the close proximity with the optimum. $\Gamma^- = [d(\{z^-\}_{k^-}) - O^-]_+$ and $\Gamma^+ = [O^+ - d(\{z^+\}_{k^+})]_+$, where O^- and O^+ represents the *expected* optimums of $d(\{z^-\}_{k^-})$ and $d(\{z^+\}_{k^+})$. Note that, we further propose a *variation* of Γ including Γ^- and Γ^+, and the comparisons of them are demonstrated in Section 4.4. γ^+ and γ^- is used to replace λ and add Γ^- and Γ^+ in Equation 8:

$$\begin{aligned}\mathcal{L}_{OUCL} = \frac{1}{\beta} log\Bigg\{1 + \sum_{k^-=1}^{K^-}\sum_{k^+=1}^{K^+} exp\Big[\beta\Big(&\Gamma^- \\ (d(\{z^-\}_{k^-}) - \gamma^-) - \Gamma^+(d(\{z^+\}_{k^+}) - \gamma^+)\Big)\Big]\Bigg\}.\end{aligned} \tag{9}$$

We limit $d(\{z^-\}_{k^-})$ and $d(\{z^+\}_{k^+})$ in the range of $[0, 1]$ by normalizing the features in $\{z^-\}_{k^-}$ and $\{z^-\}_{k^-}$, such that *theoretically*, the optimum of $d(\{z^-\}_{k^-})$ is 0, the optimum of $d(\{z^+\}_{k^+})$ is 1. The positive of $d(\{z^-\}_{k^-}) - O^-$ and $O^+ - d(\{z^+\}_{k^+})$ can easily be guaranteed. To cut the number of hyperparameters, we reform Equation 9 into

$$\begin{aligned}\mathcal{L}_{OUCL} = \frac{1}{\beta} log\Bigg\{1 + \sum_{k^-=1}^{K^-}\sum_{k^+=1}^{K^+} exp\Big[\\ \beta\Big((d(\{z^+\}_{k^+}) - 1)^2 + (d(\{z^-\}_{k^-}))^2 - 2\gamma^2\Big)\Big]\Bigg\}\end{aligned} \tag{10}$$

Table 1. Comparison of different methods on classification accuracy (top 1). We use *conv* and *fc* as backbones in the experiments. ‡ denotes that the methods have reduced learnable parameters (See Appedix B.1).

Model	Tiny ImageNet		STL-10		CIFAR10		CIFAR100	
	conv	fc	conv	fc	conv	fc	conv	fc
Fully supervised	36.60		68.70		75.39		42.27	
BiGAN	24.38	20.21	71.53	67.18	62.57	62.74	37.59	33.34
NAT	13.70	11.62	64.32	61.43	56.19	51.29	29.18	24.57
DIM	33.54	36.88	72.86	70.85	73.25	73.62	48.13	45.92
SplitBrain‡	32.95	33.24	71.55	63.05	77.56	76.80	51.74	47.02
SwAV	39.56 ± 0.2	38.87 ± 0.3	70.32 ± 0.4	71.40 ± 0.3	68.32 ± 0.2	65.20 ± 0.3	44.37 ± 0.3	40.85 ± 0.3
SimCLR	36.24 ± 0.2	39.83 ± 0.1	75.57 ± 0.3	77.15 ± 0.3	80.58 ± 0.2	80.07 ± 0.2	50.03 ± 0.2	49.82 ± 0.3
CMC‡	41.58 ± 0.1	40.11 ± 0.2	83.03	85.06	81.31 ± 0.2	83.28 ± 0.2	58.13 ± 0.2	56.72 ± 0.3
MoCo	35.90 ± 0.2	41.37 ± 0.2	77.50 ± 0.2	79.73 ± 0.3	76.37 ± 0.3	79.30 ± 0.2	51.04 ± 0.2	52.31 ± 0.2
BYOL	41.59 ± 0.2	41.90 ± 0.1	81.73 ± 0.3	81.57 ± 0.2	77.18 ± 0.2	80.01 ± 0.2	53.64 ± 0.2	53.78 ± 0.2
Barlow Twins	39.81 ± 0.3	40.34 ± 0.2	80.97 ± 0.3	81.43 ± 0.3	76.63 ± 0.3	78.49 ± 0.2	52.80 ± 0.2	52.95 ± 0.2
DACL	40.61 ± 0.2	41.26 ± 0.1	80.34 ± 0.2	80.01 ± 0.3	81.92 ± 0.2	80.87 ± 0.2	52.66 ± 0.2	52.08 ± 0.3
LooC	42.04 ± 0.1	41.93 ± 0.2	81.92 ± 0.2	82.60 ± 0.2	83.79 ± 0.2	82.05 ± 0.2	54.25 ± 0.2	54.09 ± 0.2
SimCLR + Debiased	38.79 ± 0.2	40.26 ± 0.2	77.09 ± 0.3	78.39 ± 0.2	80.89 ± 0.2	80.93 ± 0.2	51.38 ± 0.2	51.09 ± 0.2
SimCLR + Hard	40.05 ± 0.3	41.23 ± 0.2	79.86 ± 0.2	80.20 ± 0.2	82.13 ± 0.2	82.76 ± 0.1	52.69 ± 0.2	53.13 ± 0.2
CMC‡ + Debiased	41.64 ± 0.2	41.36 ± 0.1	83.79 ± 0.3	84.20 ± 0.2	82.17 ± 0.2	83.72 ± 0.2	58.48 ± 0.2	57.16 ± 0.2
CMC‡ + Hard	42.89 ± 0.2	42.01 ± 0.2	83.16 ± 0.3	85.15 ± 0.2	83.04 ± 0.2	86.22 ± 0.2	58.97 ± 0.3	59.13 ± 0.2
MetAug (only OUCL)‡	42.02 ± 0.1	42.14 ± 0.2	84.09 ± 0.2	84.72 ± 0.3	85.98 ± 0.2	87.13 ± 0.2	59.21 ± 0.2	58.73 ± 0.2
MetAug‡	$\mathbf{44.51 \pm 0.2}$	$\mathbf{45.36 \pm 0.2}$	$\mathbf{85.41 \pm 0.3}$	$\mathbf{85.62 \pm 0.2}$	$\mathbf{87.87 \pm 0.2}$	$\mathbf{88.12 \pm 0.2}$	$\mathbf{59.97 \pm 0.3}$	$\mathbf{61.06 \pm 0.2}$

which is derived by setting $O^+ = 1 + \gamma$, $O^- = -\gamma$, $\gamma^+ = 1 - \gamma$, and $\gamma^- = \gamma$.

3.4. Model objective

Concretely, we adopt margin-injected meta feature augmentation in the contrastive learning paradigm to achieve desired discriminative multi-view representations, and the proposed $\mathcal{L}_{MetAug}$ is incorporated to replace the conventional contrastive loss $\mathcal{L}$. The final model objective is defined as:

$$\mathcal{L}_{MetAug} = \mathcal{L}^{ori}_{OUCL} + \delta \cdot \mathcal{L}^{aug}_{OUCL} \tag{11}$$

where $\mathcal{L}^{ori}_{OUCL}$ represents the loss *NOT* including the meta augmented features, $\mathcal{L}^{aug}_{OUCL}$ represents the loss including such features, and δ is a coefficient that controls the balance between them (we perform parameter comparisons in Appendix A.4). It is worthy to note that the margin-injected regularization $\mathcal{R}_\sigma$ is only used in meta training the MAGs, i.e., $a_\omega(\cdot)$, while in regular training of encoders and projection heads, $\mathcal{R}_\sigma$ is discarded. $\mathcal{R}_\sigma$ restricts the augmented features to be informative so that such features can lead the encoder to efficiently and effectively learn discriminative representations. The training process is detailed by Algorithm 1.

4. Experiments

We benchmark our MetAug on five established datasets: Tiny ImageNet [29], STL-10 [12], CIFAR10 [29], CIFAR100 [29], and ImageNet [28]. The compared benchmark methods include: BiGAN [13], NAT [5], DIM [25], SplitBrain [52], CPC [26], SwAV [6], SimCLR [7], CMC [42], MoCo [23], SimSiam [8], InfoMin Aug. [43], BYOL [20], Barlow Twins [51], DACL [46] LooC [49], Debiased [11], Hard [35], and NNCLR [15].

4.1. Efficiently performing MetAug

Implementations. To efficiently perform CL within a restricted amount of the inputs in training, we uniformly set the batch size as 64 (see Appendix A.1 for the comparisons under different setting of batch size). For the experiments with *conv* and *fc* as the backbone networks, we adopt a network with the 5 convolutional layers in AlexNet [30] as conv and a network with further 2 fully connected layers as fc. Inspired by the backbone splitting setting of SplitBrain [52], we evenly split the AlexNet into sub-networks across the channel dimension, and each sub-network is the view-specific encoder (see Appendix B for the detailed implementation). For the experiments with ResNet-50, we directly change the encoder network to ResNet-50. All backbone encoders are not pre-trained. MetAug (*only OUCL*) is the ablation variant without margin-injected meta feature augmentation.

Given an RGB image, we convert it to the Lab image color space and split it into L and ab channels. During contrastive learning, RGB, L, and ab are used as three views

Algorithm 1 MetAug

Input: Multi-view dataset X with M views of each sample, minibatch size n, and hyperparameters α, β, δ.
Initialize The neural network parameters: θ and ϑ for view-specific encoders $f_\theta(\cdot)$ and projection heads $g_\vartheta(\cdot)$, ω for MAGs, i.e., $a_\omega(\cdot)$. The learning rates: ℓ and ℓ'.
repeat
  for t-th training iteration **do**
    Iteratively sample minibatch $\widetilde{X} = \{X_i\}_{i=(t-1)n}^{tn}$.
    $\#$ *regular contrastive training step*
    $\theta \leftarrow \theta - \ell \Delta_\theta \mathcal{L}_{MetAug}(f_\theta, g_\vartheta, a_\omega, \widetilde{X})$
    $\vartheta \leftarrow \vartheta - \ell \Delta_\vartheta \mathcal{L}_{MetAug}(f_\theta, g_\vartheta, a_\omega, \widetilde{X})$
  end for
  for t-th training iteration **do**
    Iteratively sample minibatch $\widetilde{X} = \{X_i\}_{i=(t-1)n}^{tn}$.
    $\#$ *compute fast weights*
    $\#$ *retain computational graph*
    $\mathring{\theta} = \theta - \ell \Delta_\theta \mathcal{L}_{MetAug}(f_\theta, g_\vartheta, a_\omega, \widetilde{X})$
    $\mathring{\vartheta} = \vartheta - \ell \Delta_\vartheta \mathcal{L}_{MetAug}(f_\theta, g_\vartheta, a_\omega, \widetilde{X})$
    $\#$ *meta training step using second derivative*
    $\omega \leftarrow \omega - \ell' \Delta_\omega \left(\mathcal{L}_{MetAug}(f_{\mathring{\theta}}, g_{\mathring{\vartheta}}, a_\omega, \widetilde{X}) + \alpha \cdot \mathcal{R}_\sigma \right)$
  end for
until θ, ϑ, and ω converge.

Table 2. Performance (accuracy) on the CIFAR10 and STL-10 datasets with ResNet-50 [24].

Model	CIFAR10	STL-10	Average
SwAV	83.15	82.93	83.04
SimCLR	84.63	83.75	84.19
CMC	86.10	86.83	86.47
BYOL	87.14	87.56	87.35
Barlow Twins	85.84	86.02	85.93
DACL	86.93	88.11	87.52
LooC	87.80	88.62	88.21
SwAV + Hard	83.99	84.51	84.25
SimCLR + Hard	86.91	85.48	86.20
CMC + Hard	88.25	87.79	88.02
MetAug (*only OUCL*)	88.79	88.31	88.55
MetAug	**91.09**	**90.26**	**90.68**

of the image. Before feeding the views into our model, we simply adopt the same data augmentations in CMC [42]. Especially, the major contribution of DACL is the proposed data augmentation (i.e., mixup-noise) so that we particularly add mixup data augmentation for DACL. In training, a memory bank [48] is adopted to facilitate calculations. We retrieve 4096 past features from the memory bank to derive negatives. The learning rates and weight decay rates are uniform over comparisons.

Comparison on downstream tasks. We collect the results of 20 trials for comparisons. The average result of the last 20 epochs is used as the final result of each trial, and the 95% confidence intervals are also reported, while the results without 95% confidence intervals are quoted from the published papers. We compare MetAug against a fully-supervised method (similar to AlexNet [30]) and the state-of-the-art unsupervised methods. Table 1 shows the comparisons on four benchmark datasets. The last two rows of tables represent the results of our methods. As demonstrated in tables, MetAug beats the best prior methods on all datasets. Even compared with the fully-supervised method trained end-to-end (without fine-tuning) for the architecture presented, the proposed method has a significant improvement on most downstream tasks, which demonstrates that MetAug can better model discriminative information when supervision is insufficient (e.g., the training data is limited). The ablation model (i.e., MetAug (*only OUCL*)) outperforms most unsupervised methods but falls short of the performance of MetAug. Thus, the ablation study proves the effectiveness of our proposed margin-injected meta feature augmentation and optimization-driven unified contrast.

DACL and LooC propose to enhance contrastive learning from the perspective of *data* augmentation, while MetAug improves contrastive learning from the perspective of *feature* augmentation. The idea behind our method is simple but effective, since contrastive learning works directly on features, and the augmented images need one step of encoding to become features. The experimental results support that

Table 3. Linear evaluation results on ImageNet. We follow the setting of [15, 42] to compare with other benchmark SSL methods with conv and ResNet-50.

ImageNet			
Model	conv	ResNet-50	
	top 1		top 5
Fully supervised	50.5	-	-
SplitBrain	32.8	-	-
CPC v2	-	63.8	85.3
SwAV	38.0 ± 0.3	71.8	-
SimCLR	37.7 ± 0.2	71.7	-
CMC	42.6	66.2	87.0
MoCo	39.4 ± 0.2	71.1	-
SimSiam	-	71.3	-
InfoMin Aug.	-	73.0	91.1
BYOL	41.1 ± 0.2	74.3	91.6
Barlow Twins	39.6 ± 0.2	-	-
NNCLR	-	75.4	92.3
DACL	41.8 ± 0.2	-	-
LooC	43.2 ± 0.2	-	-
SimCLR + Debiased	38.9 ± 0.3	-	-
SimCLR + Hard	41.5 ± 0.2	-	-
MetAug + CMC	$\mathbf{45.1 \pm 0.2}$	-	-
MetAug + NNCLR	-	**76.0**	**93.2**

Table 4. Comparison of applying benchmark SSL methods with different data augmentations by using the fc backbone on CIFAR10.

ID	Data augmentations						Methods		
	horizontal flip	rotate	random crop	random grey	color jitter	mixup	DACL	LooC	MetAug
1	✓	✓					-	80.73	**87.05**
2			✓				-	81.16	**87.53**
3				✓			-	80.70	**86.81**
4					✓		-	81.64	**87.79**
5	✓		✓				-	82.05	**88.12**
6		✓			✓		-	82.16	**88.01**
7	✓		✓			✓	80.87	82.21	**88.22**
8	✓	✓	✓	✓	✓	✓	82.09	83.17	**88.65**

MetAug achieves better performance on benchmarks.

Performing MetAug on ResNet. We perform classification comparisons on the CIFAR10 and STL-10 by using ResNet-50. Table 2 shows that MetAug and the ablation variant outperform the compared methods, which indicates that MetAug has strong adaptability to different encoders.

4.2. Benchmarking MetAug on ImageNet

Implementation. To comprehensively understand the performance of our proposed MetAug, we conduct comparisons on ImageNet and make fair comparisons with benchmark methods. The backbone encoder is conv or ResNet-50, and the results are demonstrated in Table 3. MetAug is a decoupled approach so that we can introduce MetAug in the learning paradigm of state-of-the-art to improve the performance, e.g, for the experiments using conv or ResNet-50, we perform MetAug in CMC or NNCLR, respectively.

Results. As shown in Table 3, we find that MetAug can effectively promote the performance of benchmark methods in the comparisons using both conv and ResNet-50. The results support that our proposed meta feature augmentation can enable different encoders to model discriminative information even in the large-scale dataset.

4.3. Is MetAug robust for data augmentation?

To illustrate the impacts of different data augmentations, we conducted multiple comparisons on CIFAR10 shown in Table 4. Note that horizontal flip and rotate are similar, and we use them together in the 1-th comparison. In the 5-th comparison, we take the same data augmentations as the setting of comparisons in Section 4.1. The data augmentations adopted in the 6-th comparison are as same as the setting of LooC [49]. Additionally, *mixup* is proposed by DACL [46].

We observe from Table 4 that MetAug outperforms the compared methods in all comparisons. It is worth noting that even using weak data augmentation degenerates the performance of our method as well as benchmark methods, but the performance degeneration of our method is minimal compared to others, e.g., from 8-th and 1-th comparison, we find that the gap of MetAug is 1.60%, while that of LooC is 2.44%. The results support that MetAug is robust for various data augmentations.

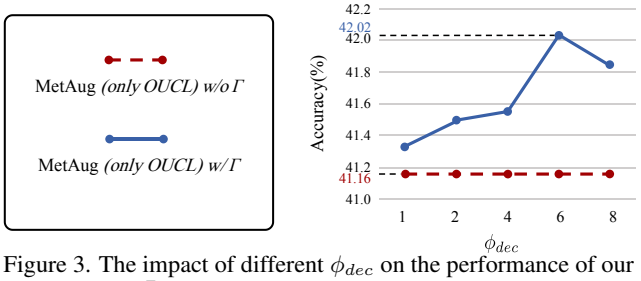


Figure 3. The impact of different ϕ_{dec} on the performance of our method using $\bar{\Gamma}$. Comparisons are conducted on Tiny ImageNet with conv as the encoders.

4.4. Do the variant of Γ promote MetAug?

In practice, we find that the introduction of the weighting factor Γ cannot directly improve our proposed method. Our conjuncture lies in that Γ may cause the loss to converge excessively fast, which leaves the network parameters at a local minimum. Therefore, we propose a variant to replace Γ in Equation 9, i.e., $\bar{\Gamma} = \frac{\Gamma}{\phi_{dec}}$ where ϕ_{dec} is a linear attenuation coefficient to linearly attenuate the impact of Γ so that the difference between the current value and the optimum becomes smaller.

We use MetAug (*only OUCL*) to demonstrate the effectiveness of the proposed variant. The results are shown in Figure 3. We observe that the performance of our method get peak value when ϕ_{dec} is 6, which manifests that introducing a certain linear attenuation to Γ can promote MetAug.

5. Conclusion

We conclude that exploring informative features is the key to contrastive learning. Different from the conventional contrastive methods that collect enough informative features to learn a good representation by enlarging the batch or

memory bank, we motivate MetAug to learn a discriminative representation from a restricted amount of images. Our method proposes margin-injected meta feature augmentation to straightforwardly augment features to be informative and avoid learning degenerate features. To efficiently make use of all available features, MetAug further proposes optimization-driven unified contrast. Experimental evaluations demonstrate that MetAug achieves the state-of-the-art.

References

[1] Succ a, Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule. 2002.

[2] S. Arora, H. Khandeparkar, M. Khodak, O. Plevrakis, and N. Saunshi. A theoretical analysis of contrastive unsupervised representation learning. 2019.

[3] Philip Bachman, R. Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *NeurIPS 2019*, 2019.

[4] Y. Bengio, S. Bengio, and J. Cloutier. Learning a synaptic learning rule. In *Ijcnn-91-seattle International Joint Conference on Neural Networks*, 2002.

[5] Piotr Bojanowski and Armand Joulin. Unsupervised learning by predicting noise. *arXiv preprint arXiv:1704.05310*, 2017.

[6] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. 2020.

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[8] X. Chen and K. He. Exploring simple siamese representation learning. 2020.

[9] Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, M. Botvinick, and N De Freitas. Learning to learn without gradient descent by gradient descent. 2016.

[10] S. Chopra, R. HAdSell, and Y. Lecun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005.

[11] C. Y. Chuang, J. Robinson, Y. C. Lin, A. Torralba, and S. Jegelka. Debiased contrastive learning. 2020.

[12] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011.

[13] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[14] Marthinus Christoffel du Plessis, Gang Niu, and Masashi Sugiyama. Analysis of learning from positive and unlabeled data. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 2014.

[15] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. 2021.

[16] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*. ACM, 2008.

[17] A. Ermolov, A. Siarohin, E. Sangineto, and N. Sebe. Whitening for self-supervised representation learning. 2020.

[18] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, Proceedings of Machine Learning Research, pages 1126–1135. PMLR, 2017.

[19] J. Goldberger, S. Gordon, and H. Greenspan. An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures. In *IEEE International Conference on Computer Vision*, 2003.

[20] J. B. Grill, F. Strub, F Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, and M. G. Azar. Bootstrap your own latent: A new approach to self-supervised learning. 2020.

[21] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. 2010.

[22] R. Hadsell, S. Chopra, and Y. Lecun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 2006.

[23] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, 2016.

[25] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[26] OJ Hénaff, A. Srinivas, J De Fauw, A. Razavi, C. Doersch, Sma Eslami, and Avd Oord. Data-efficient image recognition with contrastive predictive coding. 2019.

[27] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, and K Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. 2016.

[28] D. Jia, D. Wei, R. Socher, L. J. Li, L. Kai, and F. F. Li. Imagenet: A large-scale hierarchical image database. *Proc of IEEE Computer Vision and Pattern Recognition*, 2009.

[29] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[30] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[31] Z. Li, F. Zhou, C. Fei, and L. Hang. Meta-sgd: Learning to learn quickly for few-shot learning. 2017.

[32] S. Liu, Andrew J Davison, and E. Johns. Self-supervised generalisation with meta auxiliary learning. 2019.

[33] C. Ma, C. Shen, A. Dick, Q. Wu, P. Wang, Avd Hengel, and I. Reid. Visual question answering with memory-augmented networks. *IEEE*, 2018.

[34] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[35] J. Robinson, C. Y. Chuang, S. Sra, and S. Jegelka. Contrastive learning with hard negative samples. 2020.

[36] J Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 2014.

[37] Florian Schroff, Dmitry Kalenichenko, and James Philbin. *Facenet: A unified embedding for face recognition and clustering*. 2015.

[38] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017.

[39] K. Sridharan and S. M. Kakade. An information theoretic framework for multi-view learning. *Conference on Learning Theory*, 2008.

[40] E. Strubell, A. Ganesh, and A. Mccallum. Energy and policy considerations for deep learning in nlp. 2019.

[41] Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. *Circle loss: A unified perspective of pair similarity optimization*. 2020.

[42] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

[43] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola. What makes for good views for contrastive learning. 2020.

[44] Yhh Tsai, Y. Wu, R. Salakhutdinov, and L. P. Morency. Self-supervised learning from a multi-view perspective. 2020.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

[46] Vikas Verma, Thang Luong, Kenji Kawaguchi, Hieu Pham, and Quoc V. Le. Towards domain-agnostic contrastive learning. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, Proceedings of Machine Learning Research. PMLR, 2021.

[47] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2016.

[48] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. *Unsupervised feature learning via non-parametric instance discrimination*. 2018.

[49] Tete Xiao, Xiaolong Wang, Alexei A. Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[50] Jingjing Xu, Wangchunshu Zhou, Zhiyi Fu, Hao Zhou, and Lei Li. A survey on green deep learning. *CoRR*, 2021.

[51] J. Zbontar, J. Li, I. Misra, Y Lecun, and S. Deny. Barlow twins: Self-supervised learning via redundancy reduction. 2021.

[52] Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. 2017.

[53] Yabin Zhang, Hui Tang, and Kui Jia. Fine-grained visual categorization using meta-learning optimization with sample selection of auxiliary data. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VIII*, Lecture Notes in Computer Science, pages 241–256. Springer, 2018.

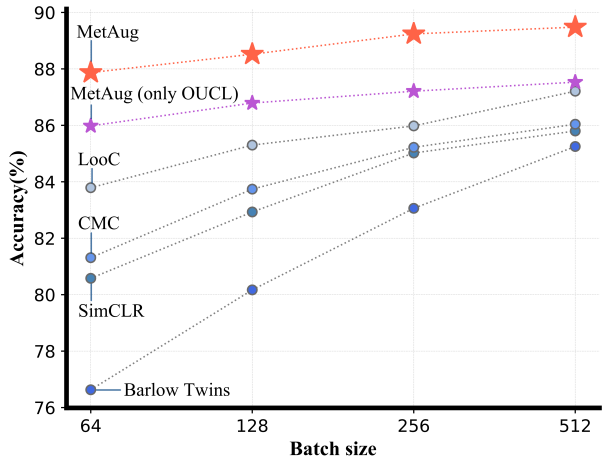


Figure 4. Comparison of different methods on classification accuracy (top1) under various settings of batch size. We conducted experiments on CIFAR10 with conv encoder.

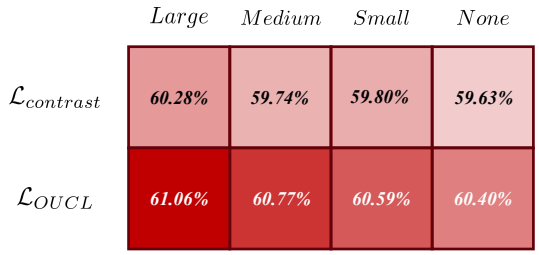


Figure 5. Heatmap of injected margin variant comparisons.

Table 5. Performance (accuracy) of MetAug with or without the augmented features on CIFAR10 with conv encoder.

Model	δ	w/ augmented features	w/o augmented features
SimCLR	-	80.58	
DACL	-	81.92	
LooC	-	83.79	
CMC + Hard	-	83.04	
MetAug	10^{-1}	**85.85**	85.48
	10^{-2}	85.91	**85.99**
	10^{-3}	86.57	**86.65**
	10^{-4}	**87.42**	87.41
	10^{-5}	87.72	**87.87**
	10^{-6}	87.26	**87.47**
	10^{-7}	86.90	**87.19**
	10^{-8}	86.12	**86.35**

A. Appendix - Extended comparisons

In this section, we provide several experimental analyses about the advantages of our proposed method. The experiments to find appropriate hyperparameters are conducted as well, and in detail, we conduct comparisons of using different hyperparameters on the validation set of corresponding benchmark datasets.

A.1. Can MetAug perform consistently under different settings of batch size?

As the results shown in Table 1, 2, and 3, we observe that MetAug achieves our expectation that learning anti-collapse and discriminative representations from a restricted amount of images in a training step (i.e., the batch size is limited). However, we conduct further experiments to explore whether MetAug has consistent performance under settings of larger batch sizes.

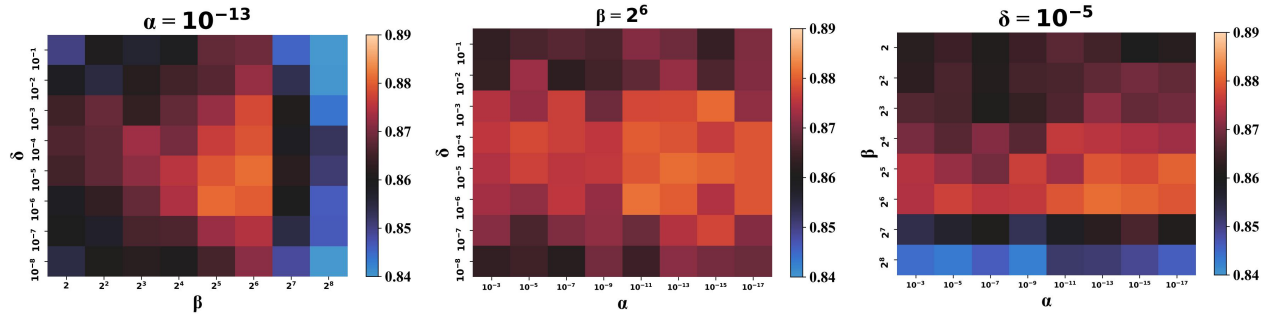
From Figure 4, we observe that with the increase of batch size, each compared method achieves better performance on the downstream task. We conjecture that with the enlarging of batch size, the number of available features in a training step is increased, so that models may explore more informative features to promote the performance of contrastive learning. Yet, comparing our method with the benchmark methods, we find that the gap between the performance of MetAug (*only OUCL*) and the compared methods becomes smaller. We extend the mentioned conjecture: as more informative features can be explored by all methods in a training step, OUCL's advantage becomes less significant. OUCL aims to include all available features to efficiently train the model and avoid the optimization fall into a local optimum, and the increase of batch size, which means sufficient self-supervision, can naturally promote the efficiency of optimization and avoid the fall into a local optimum. Yet the advance of OUCL is always maintained, which is supported by the comparison. Only LooC's performance can gradually catch up with the performance of MetAug (*only OUCL*). We research the setting of LooC, and find that LooC leverages more than one (e.g., three) contrastive loss in a training step, which allows LooC to train the model multiple times. We observe that, even in the large batch size, MetAug can still improve the state-of-the-art methods by a significant margin.

Concretely, MetAug maintains its superiority over compared method under different settings of batch size.

A.2. Variants of the injected margin

We denote $\min^+$ as $\min(\{d(\{z^+\}_{k^+})\})$ and $\max^-$ as $\max(\{d(\{z^-\}_{k^-})\})$. For *"Medium"*, both σ^+ and σ^- equal to $mean[\min^+, \max^-]$. $\sigma^+ = \max[\min^+, \max^-]$ and $\sigma^- = \min[\min^+, \max^-]$ in *"Small"*.

In Figure 5, we conduct comparisons on CIFAR100 with fc. We observe that, whether our method uses $\mathcal{L}_{contrast}$ or the proposed $\mathcal{L}_{OUCL}$, all three variants can improve MetAug, and our method with *"Large"* achieve the best performance. The experiments further prove the effectiveness of the two key ingredients of MetAug.



(a) Fix α, and study on β and δ **(b) Fix β, and study on α and δ** **(c) Fix δ, and study on α and β**

Figure 6. Impacts of the hyperparameters α, β, and δ of our proposed method. We conducted comparisons based on MetAug on CIFAR10 with fc encoder. To measure the influences, we iteratively fixed one parameter and then study on the others by selecting them in the ranges, respectively.

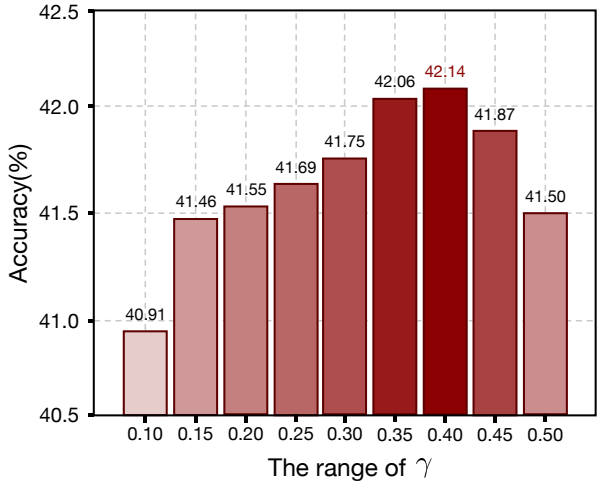


Figure 7. Impacts of the hyperparameter γ of our proposed method. We conducted comparisons based on MetAug (*onlu OUCL*) on Tiny ImageNet with fc encoder. To measure the impact of γ, we iteratively select γ and observe the accuracy of the method.

A.3. Understanding of the augmented features

To understand the augmented features, we conduct a comparison of MetAug by adopting the augmented features in the test or not. As shown in Table 5, the results of MetAug using the augmented features in the test are listed in the *w/ augmented features* column, and the results of MetAug NOT using the augmented features in the test are listed in the *w/o augmented features* column (which is the regular approach in the test). We select δ from the range of $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$ to generate different augmented features. Specifically, the approach of adopting the augmented features in the test is that we use MAGs to generate augmented features and such features are treated as the same as the original features, i.e., augmented features are regarded as additions of original features. Note that, in regular test (i.e., w/ augmented features), we use the representation h_i^j and discard the projection head $g_{\vartheta_j}(\cdot)$ to feed into the classifier, while, in the test of using augmented features, we have to use the feature z_i^j generated by the projection head $g_{\vartheta_j}(\cdot)$ to feed into the classifier, because MAGs work on the feature z_i^j.

From Table 5, we observe that, generally, MetAug *w/o augmented features* beats MetAug *w/ augmented features*. The reasons behind such phenomenon are: 1) the augmented features are generated to lead the encoders to learn discriminative representations (e.g., h_i^j), which indicates that the augmented features contribute to the improvement of the encoders, but this does not mean that the augmented features are discriminative for downstream tasks; 2) in the test of using augmented features, we do not discard the projection head $g_{\vartheta_j}(\cdot)$, and recent works prove that the approach of using a projection head in training and discarding such head in the test can significantly improve the performance of the model on downstream tasks [7, 23].

Out of the understanding of the experimental results, we think that the augmented features contain useful information that can improve the encoder, but such information may not be discriminative to downstream tasks.

A.4. Synthetic comparison of hyperparameters

To intuitively understand the impacts of hyperparameters, we conduct comparisons by using various combinations of them for the proposed MetAug. Specifically, α controls the impact of the proposed margin-injected regularization term. The hyperparameter β is proposed as a temperature coefficient in OUCL. γ is a specific parameter to replace the hyperparameters in OUCL such that the number of hyperparameters can be reduced. δ balances the impact of OUCL that uses augmented features and OUCL that does not use these features.

As demonstrated in Figure 7, we first solely study γ's impact on MetAug, because γ is only used in OUCL function, and in practice, we find that, compared with other hyperparameters, γ has less impact on our method. We conduct experiments on Tiny ImageNet with fc encoder and select γ from the corresponding range for MetAug (*only OUCL*)

to clarify its impact, and the results indicate that appropriate selected γ can indeed promote the performance of our method, but the differences between the impacts of different γ are limited.

Then, we fix $\gamma = 0.40$ and study on the impacts of other hyperparameters. As the results are shown in Figure 6, the plots further elaborate our parameter studies' results with MetAug on the CIFAR10 benchmark dataset with fc encoder. To explore the influence of β and δ, we first fixed $\alpha = 10^{-13}$, and then we selected β from the range of $\{2, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8\}$ and δ from the range of $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$. Following the same experimental principle as above, we selected α from the range of $\{10^{-3}, 10^{-5}, 10^{-7}, 10^{-9}, 10^{-11}, 10^{-13}, 10^{-15}, 10^{-17}\}$. See Figure 6(a), (b), and (c) for the details of the comparisons. In general, good classification performance highly depends on the β and δ terms. Also, α is an intensely necessary supplement for adapting the interval between similarities of augmented features and original features, which avoids to learn degenerate representations. We also find that the potential to improve the learned representations grows with the adjustment of term β, e.g., the initial loss becomes relatively large.

B. Appendix - Implementation

In this paper, we introduce a novel self-supervised representation learning approach, i.e., ***Met**a Feature **Aug**mentation* (MetAug), of which Figure 1 depicts the overview framework. The following subsections provide the design details of MetAug.

B.1. Network architecture

In the experiments, neural network classification methods (i.e., conv and fc) are adopted as the backbone networks, and the classifiers (i.e., the linear networks) on the representations extracted from the encoders are performed on downstream classification tasks.

According to the principle of building the encoders, the AlexNet is split across the channel dimension with the idea that split-AlexNet can also perform well in learning representations between views, which only has the halved learnable parameters [52]. We build the AlexNet with 5 convolutional layers, 2 linear layers, and a fully connected layer followed by a l2 normalization function. Then the split-AlexNets (i.e., the sub-networks) are regarded as the encoders. In experiments, we use conv and fc, which use the corresponding layers of AlexNet. Note that we split AlexNet across channels for RGB, L, and ab views. in the test, we concatenate representations layer-wise from the encoders into one to achieve the final representations of the inputs.

We develop the classifier by leveraging a linear network followed by a softmax output function. Following the proposed experimental setting of the previous literature [2, 25, 34, 42], we evaluate the quality of the learned representations by freezing the weights of backbone encoders and training the linear classifier in the test.

B.2. Algorithm description

MetAug is an end-to-end representation learning method: we iteratively train the encoders and MAGs by back-propagating $\mathcal{L}_{MetAug}$, and the training process is based on Adam gradient optimization.

The proposed MetAug is a generalized approach, which can be used for various downstream tasks, e.g., classification, clustering, regression, etc. We can straightforwardly train the encoders, pretrained by MetAug, on downstream tasks.

Here, we provide a pseudo-code for MetAug training described in the style of PyTorch, which is without the inclusion of the detailed matrix processing or helper utility functions & codes that are irrelevant to the algorithm:

```
# inputs: input data in a batch
# index: index for each sample (used for memory bank)
# model: view-specific encoders and projection heads
# optimizer: optimizer for model
# contrast: performing d()
# criterion: performing OUCL
# l_mtgen, ab_mtgen, ori_mtgen: view-specific MAGs
# l_mtgen_op, ab_mtgen_op, ori_mtgen_op: optimizer for MAGs
model = MyAlexNetCMC() # split AlexNet
l_mtgen = MyMetaGenNet() # 3 fully-connected layers
ab_mtgen = MyMetaGenNet() # 3 fully-connected layers
ori_mtgen = MyMetaGenNet() # 3 fully-connected layers
contrast = Contrast() # d(), and memory bank only store original features
criterion = OUCL() # performing OUCL
# Step 1: Fix metagen aug, and optimize main_model
for param in l_mtgen.parameters():
    param.requires_grad = False
for param in ab_mtgen.parameters():
    param.requires_grad = False
for param in ori_mtgen.parameters():
    param.requires_grad = False
for param in model.parameters():
    param.requires_grad = True
# ====================forward=====================
optimizer.zero_grad()
feat_l, feat_ab, feat_ori = model(inputs, False)
mt_feat_l = l_mtgen(feat_l, False)
mt_feat_ab = ab_mtgen(feat_ab, False)
mt_feat_ori = ori_mtgen(feat_ori, False)
# margin-injected loss calculation
# ori loss calc
out_ab2l, ..., out_ori2ab = contrast(feat_l, feat_ab, feat_ori, index)
loss = criterion(out_ab2l, ..., out_ori2ab)
# aug loss calc
mtl_out_ab2l, ..., out_ori2ab = contrast(mt_feat_l, feat_ab, feat_ori, index)
loss += delta * criterion(mtl_out_ab2l, ..., out_ori2ab)
mtab_out_ab2l, ..., mtab_out_ori2ab = contrast(feat_l, mt_feat_ab, feat_ori, index)
loss += delta * criterion(mtab_out_ab2l, ..., mtab_out_ori2ab)
out_ab2l, ..., mtori_out_ori2ab = contrast(feat_l, feat_ab, mt_feat_ori, index)
loss += delta * criterion(out_ab2l, ..., mtori_out_ori2ab)
mt_out_ab2l, ..., mt_out_ori2ab = contrast(mt_feat_l, mt_feat_ab, mt_feat_ori, index)
loss += delta * criterion(mt_out_ab2l, ..., mt_out_ori2ab)
# margin-injection
loss += alpha * margin_injection_loss_calc(mtl_out_ab2l, ..., out_ori2ab)
# ===================backward=====================
loss.backward()
optimizer.step()
# Step 2: Fix main_model, and optimize metagen aug
for param in l_mtgen.parameters():
    param.requires_grad = True
for param in ab_mtgen.parameters():
    param.requires_grad = True
for param in ori_mtgen.parameters():
    param.requires_grad = True
# =================meta forward==================
l_mtgen_op.zero_grad()
ab_mtgen_op.zero_grad()
ori_mtgen_op.zero_grad()
feat_l, feat_ab, feat_ori = model(inputs, False)
mt_feat_l = l_mtgen(feat_l, False)
mt_feat_ab = ab_mtgen(feat_ab, False)
mt_feat_ori = ori_mtgen(feat_ori, False)
# margin-injected loss calculation
...
# meta feat generation
feat_l, feat_ab, feat_ori = model(inputs, False)
```

```
# get meta weights
fast_weights = OrderedDict((name, param) for (name, param) in l_mtgen.named_parameters())
# create_graph flag for computing second-derivative
grads = torch.autograd.grad(loss, l_mtgen.parameters(), create_graph=True)
data = [p.data for p in list(l_mtgen.parameters())]
# compute \mathring{\theta} and \mathring{\vartheta} by applying sgd on OUCL loss
# mathcal_l: \mathcal{l}
fast_weights = OrderedDict((name, param - mathcal_l * grad) for ((name, param), grad, data) in zip(
                                                fast_weights.items(), grads, data))
mt_feat_l = l_mtgen(feat_l, fast_weights)
# get meta weights
fast_weights = OrderedDict((name, param) for (name, param) in ab_mtgen.named_parameters())
# create_graph flag for computing second-derivative
grads = torch.autograd.grad(loss, ab_mtgen.parameters(), create_graph=True)
data = [p.data for p in list(ab_mtgen.parameters())]
# compute \mathring{\theta} and \mathring{\vartheta} by applying sgd on OUCL loss
fast_weights = OrderedDict((name, param - mathcal_l * grad) for ((name, param), grad, data) in zip(
                                                fast_weights.items(), grads, data))
mt_feat_ab = ab_mtgen(feat_ab, fast_weights)
# get meta weights
fast_weights = OrderedDict((name, param) for (name, param) in ori_mtgen.named_parameters())
# create_graph flag for computing second-derivative
grads = torch.autograd.grad(loss, ori_mtgen.parameters(), create_graph=True)
data = [p.data for p in list(ori_mtgen.parameters())]
# compute \mathring{\theta} and \mathring{\vartheta} by applying sgd on OUCL loss
fast_weights = OrderedDict((name, param - mathcal_l * grad) for ((name, param), grad, data) in zip(
                                                fast_weights.items(), grads, data))
mt_feat_ori = ori_mtgen(feat_ori, fast_weights)
# margin-injected loss calculation
...
# ================meta backward==================
loss.backward()
l_mtgen_op.step()
ab_mtgen_op.step()
ori_mtgen_op.step()

class OUCL(nn.Module):
    def __init__(self):
        super(OUCL, self).__init__()
        self.criterion = nn.CrossEntropyLoss()
        self.soft_plus = nn.Softplus()
    def oucl_ite_de_to_softmax(self, sp, sn):
        # phi_dec is a hyperparameter
        ap = torch.clamp(1/phi_dec - sp.detach().div(phi_dec) + 1, max=1000, min=0.)
        an = torch.clamp(sn.detach().div(phi_dec) + 1, max=1000, min=0.)
        delta_p = 1 - gamma
        delta_n = gamma
        logit_p = - ap * (sp - delta_p) * beta
        logit_n = an * (sn - delta_n) * beta
        loss = self.soft_plus(torch.logsumexp(logit_n, dim=1) + torch.logsumexp(logit_p, dim=1)).sum().
                                                        div(sp.shape[0])
        loss = loss.div(beta).mul(16)
        return loss
    def forward(self, out_ab2l, ..., out_ori2ab):
        # \/=======calculate pos and neg========#
        pos_ab2l,neg_ab2l = torch.split(out_ab2l,[1,out_ab2l.shape[1]-1],dim=1)
        ...
        pos_ab2l = pos_ab2l.squeeze(2).add(1).div(2)
        ...
        pos = torch.cat((pos_ab2l, pos_l2ab, pos_ori2l, pos_l2ori, pos_ab2ori, pos_ori2ab), dim=1)
        neg = torch.cat((neg_ab2l, neg_l2ab, neg_ori2l, neg_l2ori, neg_ab2ori, neg_ori2ab), dim=1)
        loss = self.oucl_ite_de_to_softmax(pos, neg).sum()
        return loss
```

```
def margin_injection_loss_calc(mtl_out_ab2l, ..., out_ori2ab):
    # split pos neg
    mtl_pos_ab2l, mtl_neg_ab2l = torch.split(mtl_out_ab2l, [1, mtl_out_ab2l.shape[1]-1], dim=1)
    ...
    pos_ab2l, neg_ab2l = torch.split(out_ab2l, [1, out_ab2l.shape[1]-1], dim=1)
    ...
    # post-process
    mtl_pos_ab2l = mtl_pos_ab2l.squeeze(2).add(1).div(2)
    ...
    mt_pos = torch.cat((mtl_pos_ab2l, ..., mtori_pos_ori2ab), dim=1)
    mt_neg = torch.cat((mtl_neg_ab2l, ..., mtori_neg_ori2ab), dim=1)
    pos_ab2l = pos_ab2l.squeeze(2).add(1).div(2)
    ...
    pos = torch.cat((pos_ab2l, pos_l2ab, pos_ori2l, pos_l2ori, pos_ab2ori, pos_ori2ab), dim=1)
    neg = torch.cat((neg_ab2l, neg_l2ab, neg_ori2l, neg_l2ori, neg_ab2ori, neg_ori2ab), dim=1)
    # get max pos min neg
    min_positive_value, min_positive_pos = torch.min(pos, dim=-1)
    max_negative_value, max_negative_pos = torch.max(neg, dim=-1)
    # margin_type: a hyperparameter
    if margin_type == 'small':
        lgamma_margin_pos, _ = torch.max(torch.cat((min_positive_value.unsqueeze(1), max_negative_value
                                                    .unsqueeze(1)), dim=1), dim=-1)
        lgamma_margin_pos = lgamma_margin_pos.unsqueeze(1)
        lgamma_margin_neg, _ = torch.min(torch.cat((min_positive_value.unsqueeze(1), max_negative_value
                                                    .unsqueeze(1)), dim=1), dim=-1)
        lgamma_margin_neg = lgamma_margin_neg.unsqueeze(1)
    elif margin_type == 'large':
        lgamma_margin_pos, _ = torch.min(torch.cat((min_positive_value.unsqueeze(1), max_negative_value
                                                    .unsqueeze(1)), dim=1), dim=-1)
        lgamma_margin_pos = lgamma_margin_pos.unsqueeze(1)
        lgamma_margin_neg, _ = torch.max(torch.cat((min_positive_value.unsqueeze(1), max_negative_value
                                                    .unsqueeze(1)), dim=1), dim=-1)
        lgamma_margin_neg = lgamma_margin_neg.unsqueeze(1)
    else:
        lgamma_margin_pos = torch.mean(torch.cat((min_positive_value.unsqueeze(1), max_negative_value.
                                                  unsqueeze(1)), dim=1), dim=1, keepdim=True)
        lgamma_margin_neg = lgamma_margin_pos
    # get margin injection loss
    loss = torch.mean(torch.clamp(mt_pos - lgamma_margin_pos, min=0))
    loss += torch.mean(torch.clamp(lgamma_margin_neg - mt_neg, min=0))
    return loss
```